Facultad de Informática
UNIVERSIDADE DA CORUÑA

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN

Gestor de noticias y vídeos corporativos

Estudiante: Jorge Medín Busto
Dirección: Mariano Javier Cabrero Canosa

A Coruña, junio de 2020.

Dedicatoria

A mi madre María

Por darme siempre todas las facilidades, ayudarme en todo lo que le pedía y apoyarme en los momentos difíciles durante toda mi etapa como estudiante.

A mi padre José

Por la calma que me transmite en momentos de estrés y recordarme siempre cuales son las cosas que de verdad son más importantes.

A mis compañeros Ismael, Marcos y Martín

Por todos los buenos ratos pasados tanto dentro como fuera de la facultad, por todas las asignaturas que hemos superado juntos, ayudándonos siempre unos a otros, y por hacer más amena la carrera.

Agradecimientos

Mi agradecimiento a Mariano, tutor del trabajo de fin de grado, por su colaboración y orientación durante la realización del trabajo.

Mi agradecimiento a todos los profesores que me han impartido clase a lo largo de la carrera, por su gran labor como docentes, y no suponer en ningún caso un problema u obstáculo para aprobar la asignatura.

Resumen

Las páginas web se han convertido en un medio muy potente, utilizado cada vez más por grandes y pequeñas empresas, para llegar a los clientes y darse a conocer de un modo más rápido, sencillo y global. En estos portales web, se muestra todo tipo de contenido, ya sean noticias, listados de servicios o productos que ofrece la empresa, promociones, etc.

Este tipo de páginas se caracterizan por tener una estructura estática, donde los usuarios apenas pueden interactuar con los elementos que la forman , sino que éstos tienen una función más visual o de lectura. Uno de los componentes visuales más utilizados en los portales web es el vídeo. A través de un simple fichero o la URL de una plataforma externa, las páginas web incluyen contenido multimedia, siendo la presentación única, es decir, permitiendo sólo la difusión de un único vídeo.

El objetivo de este proyecto es realizar un gestor de vídeos y noticias, que permita crear un portal web donde se muestre este contenido y, a su vez, poder administrar los vídeos y noticias que aparecerán en el portal. La parte más compleja a desarrollar es la implementación de un reproductor en el cuál se puedan visualizar vídeos insertados a través de ficheros y también vídeos obtenidos de plataformas web externas.

Debido a la naturaleza del proyecto, se requiere la realización de varios prototipos, de manera que se le ofrezca al cliente desde un primer momento un servicio con algunos funcionamientos básicos. Por ello se usa una metodología ágil, concretamente Scrum, que se adapta a esta necesidad. Durante el desarrollo del proyecto se han elaborado diferentes versiones, siendo cada una de ellas una extensión de la realizada previamente. El reproductor de vídeo multiplataforma fue evolucionando a lo largo de todo el proceso. En la primeras versiones consistía en la unión de diferentes elementos de vídeo, los cuales se representaban en la interfaz como uno solo, mostrando uno y ocultando los demás. En las últimas versiones, el reproductor ya pasó a ser un único elemento, obteniendo el formato de carrusel deseado, el cual ya es capaz de manejar un conjunto de elementos de vídeo a través de su particular funcionamiento.

Abstract

Websites have become a very powerful media , increasingly used by large enterprises but also by the smaller ones to spread the word about their business in a faster, simple and global way. In them all kind of content is shown, from lists of services or products that the company

offers, to news and promotions.

These type of sites are characterised by a static structure where the users barely interact with their elements, which have a visual or lecture function. One of the most used components is the video element which allows to show a video file or, through an URL, a video absorbed from an external platform (such as YouTube or Vimeo), and with the restriction that only one video can be shown.

The main objective, and at the same time the most complex part to develop, of this project is to simulate the implementation of a player where inserted through files videos and those obtained from video platforms can be visualised. We say that it is going to be simulated because it does not exist any tool for websites development able to play videos from different sources.

During the project development, series of prototypes were made, being each of them an extension of the previous one. The multiplatform video player progressed throughout the process. The first versions consisted in the union of different video elements, which were represented in the interface as an only one, showing one and hiding the others. In the latest versions of the player, it became an only element, obtaining the wanted carousel format. This one is able to manage a group of video elements through its odd functioning.

Palabras clave:

- Vídeo
- Noticia
- Corporativo
- Carrusel
- Ticker
- Vimeo
- Youtube
- Empresa

Keywords:

- Video
- New
- Corporate
- Carousel
- Ticker
- Vimeo
- Youtube
- Company

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introdución

1.1 Motivación

El comercio y las ventas por Internet aumentan día a día, a un ritmo exponencial. De hecho, existen ya muchas empresas que solo ofrecen servicios online a través de sus páginas web y no cuentan con tiendas físicas. Es por ello que la gran mayoría de las empresas hoy en día ya han dado el salto a Internet, lo cual les abre muchas oportunidades de negocio. Además de ser un medio muy eficaz, es muy eficiente, ya que suponen poco gasto económico para las empresas.

En los portales web solemos encontrar los servicios o productos que ofrece la empresa al cliente, como por ejemplo, cuando realizamos una compra a través de Amazon, donde se nos muestra el conjunto de productos que se venden en esta web. Sin embargo, otras empresas usan estas páginas no solo para mostrar lo anterior, si no que muestra otro tipo de contenido. Este contenido pueden ser vídeos corporativos, noticias relacionadas con la empresa, clientes para los que se trabaja, proyectos realizados, etc. La página web sirve entonces como mecanismo autopublicitario a la empresa. Un aspecto colateral es la posibilidad de difundir noticias corporativas en tiempo real, a modo de teletipo, que pueden ser posteriormente consultadas por los clientes. De esta forma, se complementa a la información textual, reservada para lectura profunda, con la información eminentemente visual, mucho más ligera.

En publicidad como en muchos ámbitos de negocio, una imagen vale más que mil palabras. Y si está en movimiento muchos más. Desde hace tiempo el lenguaje HTML permite la inserción de vídeos en páginas web para su reproducción, ya sean almacenados en el propio servidor o embebidos de plataformas web externas. Sin embargo, estas elementos no son capaces de reproducir vídeos de orígenes distintos de manera simultánea. Como veremos a continuación, uno de los objetivos principales es implementar un reproductor que sea capaz

de hacer esto. Esto supone una motivación para el desarrollador, ya que no se trata de algo que ya había sido implementado anteriormente, si no que se las debe ingeniar para elaborar una propia implementación.

1.2 Objetivos

El proyecto consiste en el desarrollo de una plataforma web de gestión de autopublicidad de vídeos corporativos y a la difusión de noticias de la empresa en un portal web público.

El objetivo principal es que sea capaz de almacenar y visualizar distintos tipos de vídeos, ya sean vídeos almacenados en el propio servidor, como vídeos enlazados de plataformas externas. Dentro de los vídeos almacenados en el servidor, se distinguen también una amplia lista de formatos de fichero. En el portal web, debe conseguirse que en el reproductor se visualice cualquier vídeo, tratando, en lo máximo posible, que no se aprecie diferencia entre un tipo u otro. Se marca este objetivo como prioritario debido a que es la novedad que presenta el proyecto y la parte que más problemas plantea. Del mismo modo que los vídeos, debe permitir almacenar y visualizar las noticias.

Otro objetivo imprescindible es que la plataforma web tenga un diseño Responsive, es decir, que la interfaz sea capaz de adaptarse al dispositivo en el que se esté usando, tanto para las pantallas del portal web como de administración.

Con este proyecto se busca ofrecer nuevas alternativas de publicidad para las empresas a través de los vídeos, un medio de comunicación que es capaz de llamar rápidamente la atención del usuario, transmitiendo mucha información mediante imágenes en movimiento y sonido.

El objetivo de la empresa es hacer llegar su contenido al mayor número de usuarios posible. Por ello la plataforma debe ser accesible desde cualquier dispositivo, ya sea móvil, ordenador o tablet, y que sea capaz de representar correctamente los elementos de la interfaz en cada uno de ellos para que la información llegue al usuario adecuadamente. De este modo conseguimos que los usuarios puedan acceder a la página web en cualquier momento y lugar. Además, debe adaptarse a la circunstancias individuales de cada usuario, para que este se encuentre cómodo usando la plataforma, y ofrecer soluciones para personas con discapacidades que le impidan acceder a la información.

1.3 Organización de la memoria

Herramientas y tecnologías usadas

Se enumeran las herramientas y tecnologías software usadas durante la realización del proyecto, justificando su utilización y adecuación al proyecto.

Estudio de antecedentes

Se revisan plataformas similares ya existentes, con el objetivo de identificar sus debilidades, comparándolo con las soluciones que ofrece nuestra plataforma.

Metodología

Se define la metodología usada y las razones de porque se usa, se realiza una aproximación de los tiempos y costes y se define por fases como va a ser el desarrollo del proyecto.

Diseño

Diseño de las distintas capas que van a formar el software, así como la definición de requisitos que se deben cumplir.

Implementación

Desarrollo del código de los diferentes prototipos, revisión y corrección de versiones anteriores.

Pruebas

Realización de pruebas que verifiquen y validen el correcto funcionamiento de la aplicación, tanto de aspectos funcionales como no funcionales.

Manual de Usuario

Se muestran las pantallas de la plataforma y el funcionamiento de los elementos de cada pantalla.

Conclusión

Descripción del estado final de la aplicación, los objetivos que se han logrado y funcionalidades que se pueden implementar en el futuro.

Capítulo 2

Herramientas y tecnologías utilizadas

Antes de comenzar a realizar un trabajo es importante conocer qué herramientas disponibles hay para cada aspecto del proyecto, tanto de hardware como de software. Para este proyecto no se ha necesitado ningún tipo de hardware, por lo que en este apartado únicamente se mencionan herramientas de software. A continuación, se nombran las tecnologías que se han utilizado y cual ha sido su utilidad.

En primer lugar, se ha decidido cual es el entorno de desarrollo en el que se va a trabajar. Los más populares son **Eclipse** e **Intellij Idea**. A pesar de que Intellij Idea cuenta con una versión comercial, ambos ofrecen una versión libre. Es cierto que Eclipse tiene un mejor rendimiento en cuanto a consumo de RAM. Aun así, salvo en casos de problemas a causa de los recursos del PC, el IDE no supone ninguna ventaja o inconveniente, por lo que lo más correcto es elegir cada desarrollador en el que se encuentre más cómodo y sea capaz de trabajar de un modo más fluido. Por ello, se ha elegido trabajar con Intellij Idea.

Se ha usado **GIT** para el control de versiones. Se usó con el fin de llevar un control de los cambios realizados, comparar implementaciones anteriores con la actual y crear versiones de la aplicación a través de los tags o etiquetas. Además supone una medida de seguridad, ya que en caso de ocurrir algún problema con el proyecto en local, siempre estará guardado en el repositorio. Otra utilidad empleada es la creación de ramas. Las ramas fueron usadas para implementar nuevas ideas o mejoras que sustituyen una implementación que ya está funcionando y es estable, todo esto sin temor alguno de perder la implementación actual.

Para almacenar los datos se usa una base de datos. El gestor usado es **MySQL**. Otros gestores open source como PostgreSQL ofrecen mayor fiabilidad e integridad, además de diversas

funcionalidades propias, pero esto es util para aplicaciones que realicen operaciones que manejen grandes cantidades de datos. Para aplicaciones web que no implican esa gran cantidad de datos y realizan modificaciones simples, MySQL es ideal por su sencillez y rapidez. [1]

Para la creación y elaboración del proyecto se ha usado **Spring Boot**. Este framework nos ofrece todo lo que aporta Spring, pero de un modo más ágil y a través del propio lenguaje de java, lo cual es mucho más sencillo que manejar ficheros XML.

Para la construcción y estructuración del proyecto se usa **Maven**. Maven permite dividir el proyecto en módulos, lo cual nos permite distribuir las clases y ficheros de configuración necesarios según la arquitectura del proyecto, en este caso siguiendo, el patrón MVC. También permite añadir las dependencias de las librerías de Java que sean necesaria. Por último, los IDEs cuentan con una implementación gráfica de Maven. Esto hace que su funcionamiento sea muy sencillo e intuitivo.

Para la conexión entre el modelo Java y la base de datos se usa **JPA/Hibernate**. Es un framework muy util, ya que se encarga de generar y actualizar las tablas en la base de datos por si solo. Al realizar operaciones simples (CRUD), JPA cuenta con funciones que realizan las consultas a la BD, lo que exime al desarrollador de escribir las consultas.

Para el desarrollo de la interfaz gráfica se ha usado el framework **Backbone JS**. En el resumen de esta memoria se dice que los portales web como los que se desarrollan en este proyecto son de una estructura estática y de poca interacción con el usuario. Para estos casos, el uso de Backbone no se recomienda. Sin embargo, aunque en nuestro portal web no haya mucha interacción con el usuario, si que se están produciendo eventos constantemente, ya que cuando termina un vídeo, se lanza un evento que se encarga de reproducir el siguiente vídeo. Además Backbone sigue el paradimga MVC, en concordancia con la arquitectura del proyecto. Otras de sus ventajas es que facilita el uso de vistas y permite el uso de colecciones, lo cual simplifica la parte de administración de la aplicación. Otros frameworks JS usados son **JQuery**, que hace más intuitiva la interacción con los elementos HTML, y **AJAX**, usado para realizar las peticiones al servidor y poder obtener o modificar datos desde la interfaz. Cabe destacar que para el funcionamiento del reproductor se han usado las API JS de Vimeo y YouTube.

Para la realización de pruebas se han usado las librerías de Java **JUnit** (pruebas de integración) y **Mockito** (Pruebas unitarias). Durante el desarrollo del servicio se han utilizado también aplicaciones como **Postman**, que permite crear y ejecutar peticiones REST.

Capítulo 3

Estudio de antecedentes

Existen muchos portales web como el que se realiza en el proyecto. De hecho, el objetivo de este proyecto es realizar algo similar a lo que ya existe, pero con un elemento que aporte una nueva funcionalidad, que es el reproductor multiplataforma.

La gran mayoría de las portales web de las empresas tienen una estructura muy similar:

- En la parte posterior una cabecera con el nombre/logo de la empresa y una serie de enlaces a otras páginas.
- Imagen o carrusel de imágenes relativos a la empresa
- Cuerpo de la página, que contiene noticias, ofertas de trabajo o cualquier tipo de información

Veamos a continuación una serie de ejemplos de estos portales web, a partir de los cuales se ha inspirado este proyecto:

- Gadis (www.gadis.es)

  En la página web de Gadis podemos ver como cumple con los tres elementos mencionados anteriormente (Figura 3.1). Respecto al segundo punto, cuenta con un carrusel de imágenes o gifs con deslizamiento automático entre imágenes , aunque si hacemos clic sobre una de esas imágenes, podemos que son enlaces a vídeos de Youtube. Esto conlleva que el usuario tenga que salir de nuestra interfaz. En este proyecto se implementa un carrusel que permitiría mostrar directamente los vídeos en el portal, sin necesidad de crear enlaces a ellos.

Figura 3.1: Pagina web de Gadis

- Soovil (www.soovil.com/consultar-anuncios.php)

  Soovil es una página web de publicación de anuncios. En este caso no nos centraremos en la pantalla donde se visualizan, sino en la parte de administración (Figura 3.2) . En la imagen podemos ver los anuncios, para los cuales guarda un usuario, enunciado y una duración, que indica hasta cuando se va a publicar ese anuncio. Con estos datos, parece que todo anuncio almacenado se va a mostrar hasta la fecha indicada. En nuestro proyecto, se usan dos fechas para cada vídeo y noticia, lo que nos permite almacenar un elemento nuevo en el servidor, sin que este se vaya a publicar directamente, y quede programado automáticamente para que el vídeo o noticia aparezca en el portal web en su fecha correspondiente.

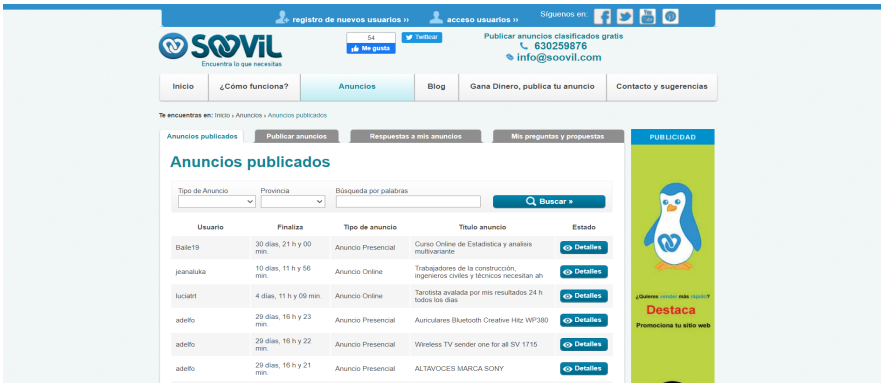

Figura 3.2: Pagina web de Soovil

Otro problema de esta página es que no se adapta al tamaño del navegador o dispositivo, de modo que es necesario realizar scroll continuamente para poder acceder a los elementos.

- Renault (https://www.renault.es/)

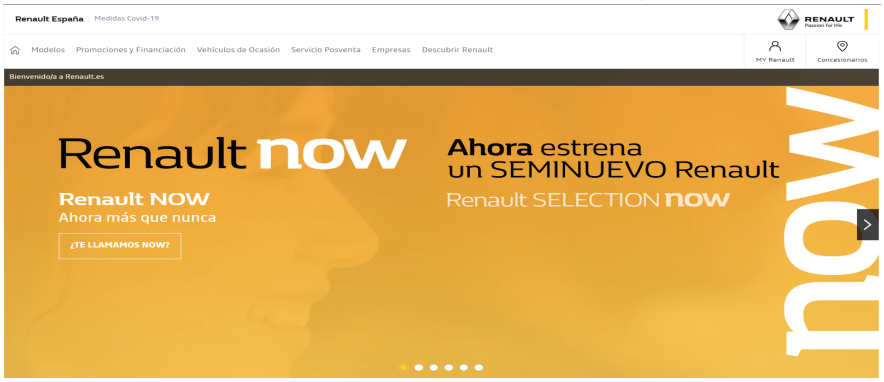

Figura 3.3: Pagina web de Renault

En la página web de Renault se usa también un carrusel como habíamos visto en la página de Gadis. La diferencia es que en este caso uno de los elementos se trata de un vídeo, el cual esta almacenado en un servidor de la empresa (Figura 3.3).

Sin embargo, en este caso, el carrusel no es capaz de deslizarse automáticamente, ya que sino se pasaría al próximo vídeo o imagen sin que éste finalizase. Otro aspecto en el que nos podemos fijar, es que el único elemento que es un vídeo es siempre el primero. Al ser un vídeo obtenido de un servidor, si inspeccionamos la página podemos ver que se usa la etiqueta de HTML <video>. Dicha etiqueta cuenta con la opción de ”autoplay”, lo cual permite reproducir el vídeo automáticamente en cuanto se tarda. Si alguno de los siguientes elementos fuese otro vídeo, si se usase el atributo autoplay nuevamente, cuando el carrusel llegase a ese vídeo, éste ya se estaría reproduciendo desde que se carga la pagina, por lo cual no se vería desde el principio. En caso de que no usase el atributo de autoplay, se debe implementar la llamada para que se reproduzca el vídeo.

Esto es algo que si se implementa en nuestro proyecto. En el carrusel todos los elementos son vídeos, y hasta que uno no termine, no se produce el efecto de deslizamiento al siguiente elemento. Además, se programa para que cada vídeo se reproduzca desde el principio cada vez que sea el elemento activo del carrusel.

En cuanto a la adaptación del elemento de vídeo, se usa un recurso muy habitual en estas páginas. Este consiste en mostrar una imagen en lugar del vídeo cuando este alcanza ciertas dimensiones reducidas.

Podemos ver que en muchos portales de empresas importantes se usa el recurso del carrusel. Sin embargo en casi todos los casos se lleva a cabo con imágenes, nunca con vídeos, y es esto lo que hace diferente nuestra plataforma. Otro aspecto positivo de nuestra aplicación es que todos los elementos de la interfaz son capaces de adaptarse al dispositivo en el que se muestran, incluido el reproductor que, independientemente del tamaño, muestra el vídeo correctamente y con unas dimensiones adecuadas.

Capítulo 4

Metodología

Las metodologías ágiles son aquellas que permiten adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno. [2]

Dada la naturaleza de la aplicación, será imprescindible disponer de distintos prototipos a lo largo del proceso de desarrollo. Debido a esto se ha usado una metodología ágil, en concreto la metodología Scrum A.1, junto con la metodología Kanban A.2, con el objetivo de desarrollar distintas versiones de la aplicación. Cada una de estas versiones supone el añadido de una nueva funcionalidad. De este modo, el desarrollo de la aplicación es progresivo y esta será funcional desde la primera versión. Ambas metodologías se describen detalladamente en el anexo.

4.1 Aplicación en el proyecto

Las dos metodologías deben adaptarse, ya que están pensadas para la realización de proyectos donde el equipo de desarrolladores esta formado por múltiples personas, existe un líder de proyecto y un cliente real. En este caso, el autor del presente trabajo ha tomado el rol de equipo de desarrollo, Scrum Master y Product Owner. Por otra parte, el rol de cliente lo asume el director del trabajo.

Las reuniones diarias se sustituyen por una documentación previa a la realización del trabajo, en la cual se anotan las tareas realizadas el día anterior o el último día de trabajo, los problemas encontrados junto con las soluciones en caso de haber dado con ellas el mismo día, y por último las tareas previstas para el día actual.

Las reuniones con el cliente se han tenido que adaptar a diversas circunstancias impre-

vistas. La primera se llevó a cabo de manera presencial y en ella el cliente explicó cuál era el problema a resolver y se definieron los requisitos básicos del proyecto. El resto de reuniones se realizaron de manera no presencial a través de videollamadas o a través del correo electrónico.

El objetivo de estas reuniones era explicar al cliente los aspectos que se han implementado y mostrar el estado en el que se encontraba la aplicación. Una vez hecho esto, el cliente indica qué aspectos se deben revisar o corregir de lo que se le ha mostrado. Por último, se definen los pasos del próximo sprint.

En cuanto a la metodología Kanban, se ha llevado a cabo un seguimiento del estado de cada una de las tareas definidas en los sprints. Para ello se ha usado la herramienta gratuita Zenkit, que permite de un modo muy sencillo la creación de tareas y los distintos estados por los que debe pasar. Además, Zenkit permite añadir diversos tipos de campos a cada tarea, asi como título, descripción, fecha de inicio, duración, etc.

Para hacer más sencillo el cálculo de costes del proyecto, se ha realizado un control de la duración que ha supuesto cada tarea.Para ello, se le ha añadido a cada una un campo denominado ”Tiempo Previsto”, donde se indica en horas el tiempo que se ha previsto para finalizar dicha tarea. Una vez finalizada, se añade un nuevo campo, indicando las horas reales que ha llevado terminarla.

En la figura 4.1 se muestra un ejemplo puntual del estado de la pizarra Kanban, donde podemos ver las tareas en de un sprint en diferentes estados. Los estados por los que pasa una tarea son los siguientes:

- Por hacer: tarea todavía sin comenzar
- En progreso: tarea que se esta implementado actualmente
- Hecho: tarea finalizada y revisada
- En revisión: tarea pendiente de revisión por el cliente, en corrección tras haber realizado una reunión con el cliente, o en caso de que hayan cambiado los requisitos

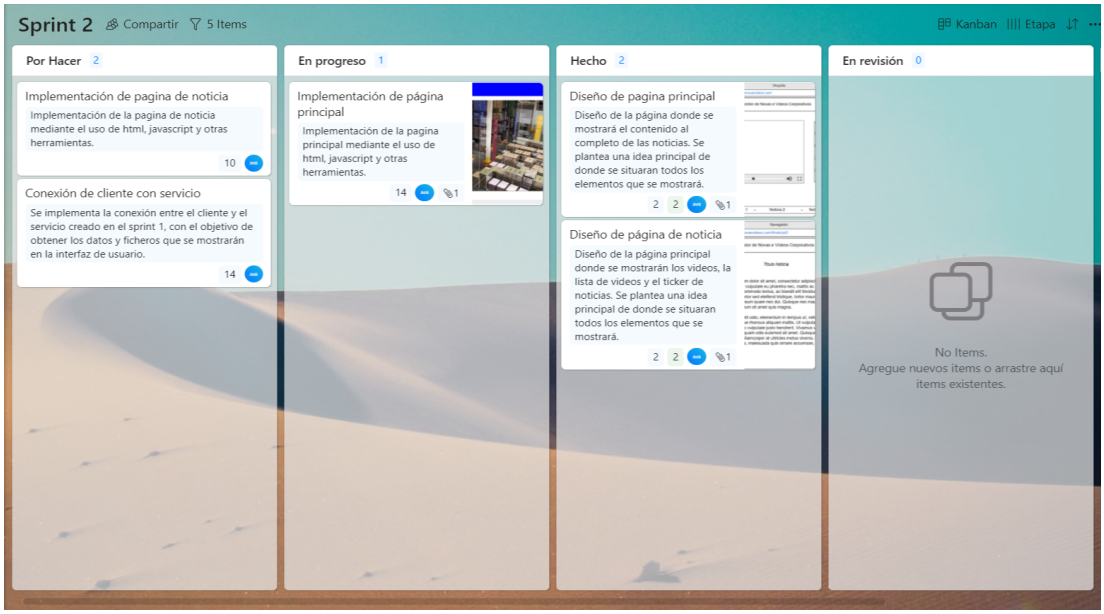

Figura 4.1: Kanban

4.2 Fases de desarrollo

En este apartado se enumeran los objetivos y las tareas principales en las que se divide cada sprint.

Sprint 1

El objetivo del sprint 1 es crear el modelo y el servicio de la plataforma, en el cual se implementan todas las operaciones relacionadas con los vídeos y las noticias. Las tareas en las que se divide son:

- 1.1 Inicio de proyecto
- 1.2 Implementación de capa modelo
- 1.3 Pruebas de capa modelo
- 1.4 Implementación del servicio web
- 1.5 Pruebas servicio web
- 1.6 Revisión y correcciones

Sprint 2

El objetivo del sprint 2 es diseñar e implementar la pantalla del portal web, en la cual se visualizan los vídeos y los titulares de las noticias, además de la vista de una noticia específica. Las tareas en las que se divide son:

- 2.1 Diseño de página inicial
- 2.2 Diseño de página de noticia
- 2.3 Implementación de página inicial
- 2.4 Implementación de página de noticia
- 2.5 Conexión de interfaz con el servicio
- 2.6 Revisión y correcciones

Sprint 3

El objetivo del sprint 3 es diseñar e implementar las pantallas de administración de la plataforma, a través de las cuales podremos realizar las operaciones del servicio a través de la interfaz, como añadir, editar o eliminar vídeos. Las tareas en las que se divide son:

- 3.1 Diseño de pantalla de administración de vídeos (listado de vídeos, pantalla de inserción y edición de vídeos)
- 3.2 Diseño de pantalla de administración de noticias (listado de noticias, pantalla de inserción y edición de noticias)
- 3.3 Implementación de pantalla de administración de vídeos
- 3.4 Implementación de pantalla de administración de noticias
- 3.5 Conexión capa cliente con capa servicio para insertar, editar o eliminar datos
- 3.6 Revisión y correcciones

Sprint 4

El objetivo del sprint 4 es implementar medidas de seguridad en la aplicación. Además de eso, se ha realizado otras tareas relacionadas con las excepciones del servicio y el uso de logs. Las tareas en las que se divide son:

- 4.1 Gestión de usuarios

- 4.2 Página de login
- 4.3 Gestión de errores en servicio e interfaz: códigos de error, validación de formularios
- 4.4 Implementar aspectos de seguridad en la aplicación
- 4.5 Uso de logs en el servicio
- 4.6 Revisión y correcciones

Sprint 5

El sprint 5 está enfocado en mejorar la interfaz de usuario. Las tareas en las que se divide son:

- 5.1 Internacionalización
- 5.2 Responsive
- 5.3 Accesibilidad y estándar W3C
- 5.4 Estilo personalizado
- 5.5 Revisión y correcciones

4.3 Costes del proyecto

En este apartado se indica un coste aproximado del proyecto. Dado que para la realización del proyecto no ha sido necesario adquirir ningún componente hardware y todas las herramientas de software usadas deben ser libres, el único coste que supone es el del desarrollador.

El cálculo se obtendrá a partir de las horas trabajadas, no por el tiempo transcurrido desde que se empezó hasta una vez terminado, ya que no ha sido posible trabajar en el proyecto todos los días el mismo número de horas, habiendo incluso días en lo que no se ha avanzado nada.

Para poder hacer un cálculo exacto se han controlado las horas de trabajo para cada uno de los pasos que forman los sprints. Además nos permite comparar el tiempo real con el tiempo que se había planificado. Podemos ver los resultados en la tabla 4.1.

El número de horas real que ha durado el proyecto es de 326. Suponemos un sueldo promedio para un desarrollador de 1500€/mes, obtenemos de un modo aproximado que su sueldo por hora, que sería de 9€/hora.

$$\text{Coste total} \approx 326 * 9 \approx 2934€$$

	Tarea	Horas previstas	Horas reales
1	Sprint 1	60	78
1.1	Inicio de proyecto	12	15
1.2	Implementación de capa modelo	10	14
1.3	Pruebas de capa modelo	12	18
1.4	Implementación del servicio web	8	14
1.5	Pruebas servicio web	8	9
1.6	Revisión y correcciones	10	8
2	Sprint 2	50	86
2.1	Diseño de página inicial	2	2
2.2	Diseño de página de noticia	2	2
2.3	Implementación de página inicial	14	30
2.4	Implementación de página de noticia	10	10
2.5	Conexión de interfaz con el servicio	14	25
2.6	Revisión y correcciones	8	18
3	Sprint 3	38	59
3.1	Diseño de pantalla de administración de vídeos	2	2
3.2	Diseño de pantalla de administración de noticias	2	1
3.3	Implementación de pantalla de administración de vídeos	10	14
3.4	Implementación de pantalla de administración de noticias	10	7
3.5	Conexión capa cliente con capa servicio	12	20
3.6	Revisión y correcciones	8	15
4	Sprint 4	48	48
4.1	Gestión de usuarios	8	12
4.2	Página de login	4	4
4.3	Gestión de errores en servicio e interfaz	8	5
4.4	Seguridad	16	22
4.5	Uso de logs en el servicio	4	1
4.6	Revisión y correcciones	8	4
5	Sprint 5	46	55
5.1	Internacionalización	12	12
5.2	Responsive	10	23
5.3	Accesibilidad y estándar W3C	12	8
5.4	Estilo personalizado	8	4
5.5	Revisión y correcciones	4	8
	Total	242	326

Tabla 4.1: Costes

Capítulo 5

Diseño

En este capítulo se muestran las especificaciones y la arquitectura que va a definir como se va a construir y estructurar el software.

5.1 Catálogo de requisitos

Los requisitos describen una necesidad del producto. Estas necesidades las podemos dividir en dos tipos: funcionales y no funcionales. La gran mayoría de los requisitos son impuestos por el cliente, el cuál tiene la obligación de comunicar y detallar en las reuniones. En este proyecto, es el cliente el que define los requisitos relacionados con las gestión de las noticias y los vídeos, y el comportamiento de los elementos en la interfaz del portal web. Además de estos requisitos, el equipo de desarrollo ha decidido imponer unos requisitos mínimos en relación a la usabilidad y la seguridad de la aplicación, a modo de buenas prácticas y con el objetivo de mejorar el servicio prestado al cliente.

5.1.1 Requisitos funcionales

Los requisitos funcionales son aquellos que describen el comportamiento o función del sistema. Los requisitos funcionales definidos son los siguientes:
Gestión de vídeos

- Los vídeos podrán cargarse/almacenarse desde el servidor/el PC del cliente y desde plataformas de vídeo.
- Los vídeos obtenidos de plataforma de vídeo deben usar toda la información posible que ofrezca la misma.
- Los vídeos obtenidos del servidor pueden ser de múltiples formatos

Gestión de noticias

- Los datos que se guardarán serán título, fecha de inicio, fecha de fin y cuerpo de la noticia.

Gestión de usuarios

- No existen usuarios como tal, sino administradores
- Solo existe un rol (administrador)
- Todos los administradores tienen acceso a todas las funcionalidades
- Todos los administradores tienen los mismos permisos y privilegios entre ellos

Interfaz

- La reproducción de vídeos debe ser automática

5.1.2 Requisitos no funcionales

Los requisitos no funcionales son las cualidades que debe tener el producto. Estos requisitos hacen que el producto sea atractivo, útil, rápido, fiable o seguro. Los requisitos no funcionales definidos son los siguientes:

Usabilidad

- Todas las pantallas cuentan con una barra de navegación para acceder a las demás pantallas
- Todas las operaciones deben poder realizarse usando únicamente el teclado
- El reproductor de vídeo permite modificar su volumen
- El reproductor de vídeo permite mostrar subtítulos

Seguridad

- Hay que iniciar sesión para acceder a las pantallas de administración
- Las contraseñas se almacenan encriptadas
- Los campos de contraseñas deben ocultar su contenido
- La aplicación debe protegerse ante los principales ataques contra aplicaciones web

Interfaz

- Debe lograr la adaptabilidad al dispositivo usado para visualizarlo (responsive)
- Los reproducción de vídeos se muestra en formato carrusel o similar
- Las noticias se visualizan en formato scroll horizontal

Software

- Todo el softaware usado para la realización del portal web debe ser libre

5.2 Arquitectura

La arquitectura del software sigue el patrón MVC, separando y abstrayendo la implementación en cada una de las capas características de este patrón de diseño.

5.2.1 Modelo

Capa de lógica de negocio

En esta capa se implementa la lógica de negocio, donde se ven reflejado los distintos casos de uso de la aplicación. Se crea una entidad para cada uno de los elementos que se gestionan en la aplicación: noticia y vídeo.

Ambas entidades cuentan con dos campos denominados Fecha de Inicio y Fecha de Fin. Estos campos indican qué vídeos y noticias se van a mostrar en cada momento en el portal web. Cada vez que éste se carga, se enviará una petición al servicio con la fecha actual, y devolverá los vídeos y las noticias para los cuales la fecha actual se encuentre entre su fecha de inicio y fecha de fin.

Tanto para vídeos como noticias se almacenan ficheros en el servidor. En caso de las noticias, se almacena un fichero que contendrá el cuerpo de la noticia y en el caso de los vídeos guardados en el servidor, se almacena un fichero para la miniatura y otra para el vídeo. Ambas clases cuentan con campos que hacen referencia a cada uno de estos ficheros, los cuales serán un String con la ruta y el nombre del fichero.

En la figura 5.1 se puede apreciar el modelado de las entidades del dominio de la aplicación, ambas cuentan con sus respectivos atributos, así como los correspondientes metodos constructores, getters/setters, equals y hashCode.

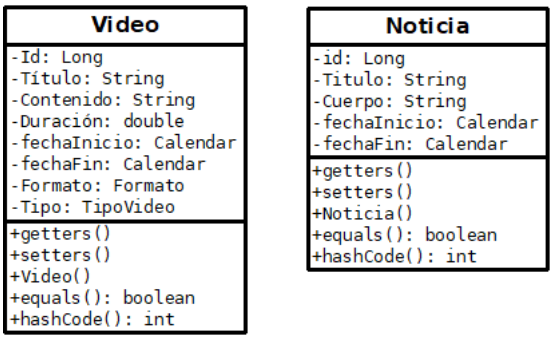

Figura 5.1: Diagramas Entidades

En la figura 5.2 podemos ver el servicio de la capa modelo, con las firmas de sus métodos, donde se implementa toda la lógica de negocio.

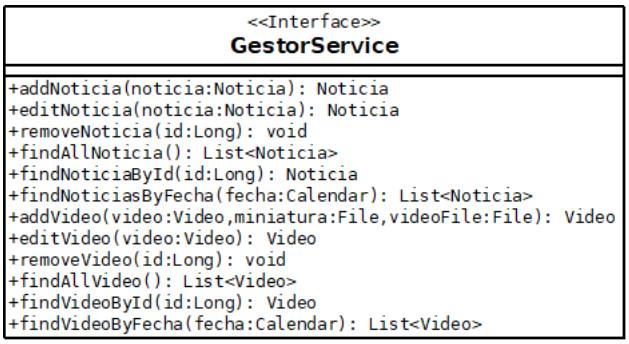

Figura 5.2: Interfaz Gestor

Capa de Acceso a datos

Para el acceso a datos se sigue el patrón DAO, el cual propone separar por completo la lógica de negocio con la lógica de acceso a datos. El DAO proporciona las llamadas funciones CRUD, lo que nos permite añadir, editar, eliminar y obtener datos en nuestra fuente de datos, que en este caso es una base de datos (Figura ??).

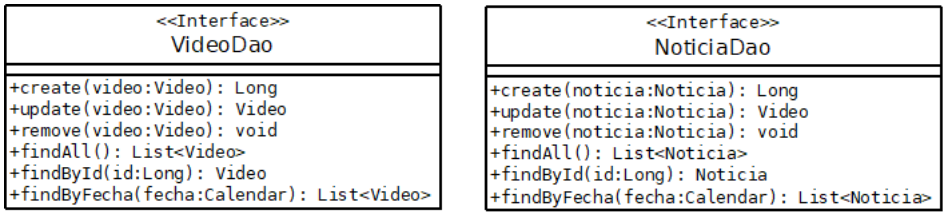

Figura 5.3: Diagramas DAOs

5.2.2 Controlador

El controlador es el intermediario entre la vista y el modelo. El controlador solicita los datos al modelo y se encarga después de enviárselos a la vista. En nuestro caso el controlador es un servicio web REST, el cual expone al cliente las operativas de la capa modelo a través de endpoints. Éstos se definen en la tabla 5.1 y la tabla 5.2

5.2.3 Vista

La vista o la interfaz de usuario, es la capa donde se muestran al usuario los datos devueltos por la capa modelo y a través de la cual el usuario interactúa con el servicio.

Las pantallas de la plataforma son el portal web, menú de administración, pantallas de gestión de vídeo y noticias, y pantalla de formularios. Para cada una de ellas se ha realizado un diseño, previo a su implementación, mediante el uso de mockups y bocetos, con el objetivo de estructurar y definir todos lo elementos necesarios para cada una de las pantallas:

- **Pantalla de Inicio**

  En la pantalla de inicio se muestra el portal web, el cual se compone de un reproductor de vídeos, una lista de imágenes y un elemento deslizable con las noticias. Dado que son muy pocos los elementos que la componen, el objetivo es que todos aparezcan en la pantalla y que la ocupen en su totalidad, sin necesidad nunca de hacer scroll para poder visualizarlos. Situado a continuación del reproductor, se muestra un texto relacionado con el vídeo que se está reproduciendo en un momento dado, como pueden ser el título y una breve descripción (Figura 5.4).

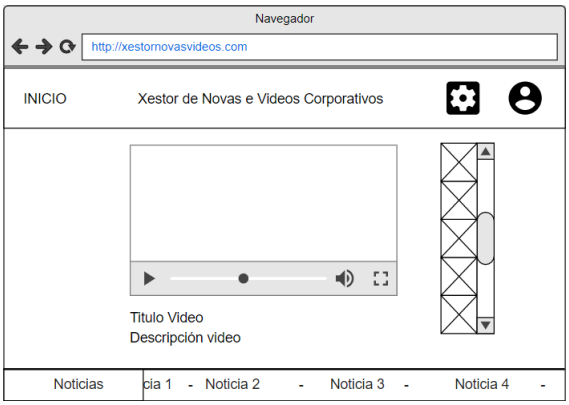

Figura 5.4: Mockup Portal Web

Nombre	Añadir Video
URL	/videos
Método	POST
Content-Type	OCTET STREAM,MULTIPART FORM DATA
Entrada	VideoDto(body) - MultipartFile(minatura) - MultipartFile(video)
Salida	VideoDto
Código	201 CREATED
Nombre	Editar Vídeo
URL	/videos/id
Método	PUT
Entrada	VideoDto
Salida	-
Código	204 No Content
Nombre	Eliminar vídeo
URL	/videos/id
Método	DELETE
Entrada	-
Salida	-
Código	204 No Content
Nombre	Buscar todos los vídeos
URL	/videos
Método	GET
Entrada	-
Salida	List<Video>
Código	200 OK
Nombre	Buscar vídeo por fecha
URL	/videos?fecha=dd-MM-yyyy
Método	GET
Entrada	-
Salida	List<Video>
Código	200 OK

Tabla 5.1: Operaciones de vídeo

Nombre	Añadir noticia
URL	/noticias
Método	POST
Entrada	NoticiaDto
Salida	NoticiaDto
Código	201 CREATED
Nombre	Editar noticia
URL	/noticias/id
Método	PUT
Entrada	NoticiaDto
Salida	-
Código	204 No Content
Nombre	Eliminar noticia
URL	/noticias/id
Método	DELETE
Entrada	-
Salida	-
Código	204 No Content
Nombre	Buscar todas la noticias
URL	/noticias
Método	GET
Entrada	-
Salida	List<NoticiaDto>
Código	200 OK
Nombre	Buscar noticias por fecha
URL	/noticias?fecha=dd-MM-yyyy
Método	GET
Entrada	-
Salida	List<NoticiaDto>
Código	200 OK

Tabla 5.2: Operaciones de noticia

- **Menú de administración** Menú simple con enlaces a las distintas pantallas de administración de la plataforma (Figura 5.5).

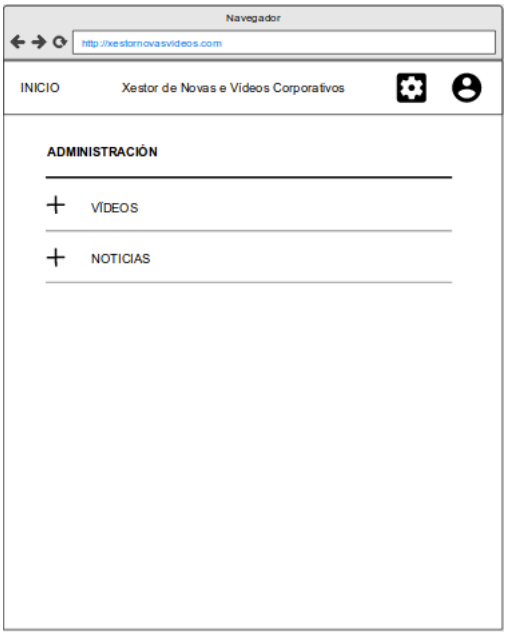

Figura 5.5: Mockup Menú Administración

- **Pantallas de administración de vídeo y noticias**

  Ambas pantallas tienen una estructura idéntica (Figura 5.6). Cuentan con una tabla que muestra todos los elementos obtenidos del servidor. Cada fila de la tabla contiene los datos de cada elemento y dos botones que permiten editar y eliminar dicho elemento.

  En la parte superior de la pantalla se encuentran los elementos de búsqueda, que permiten filtrar los datos de la tabla, y un botón que abre un formulario para añadir un nuevo elemento.

- **Formularios creación y edición de vídeos y noticias**

  Para crear o editar los elementos se muestra un formulario que contiene los campos para cada una de las entidades. Cuando se edita, los campos aparecerán completados con los valores actuales del elemento que se va a modificar. Podemos ver un ejemplo para cada caso en las figuras 5.7 y 5.8

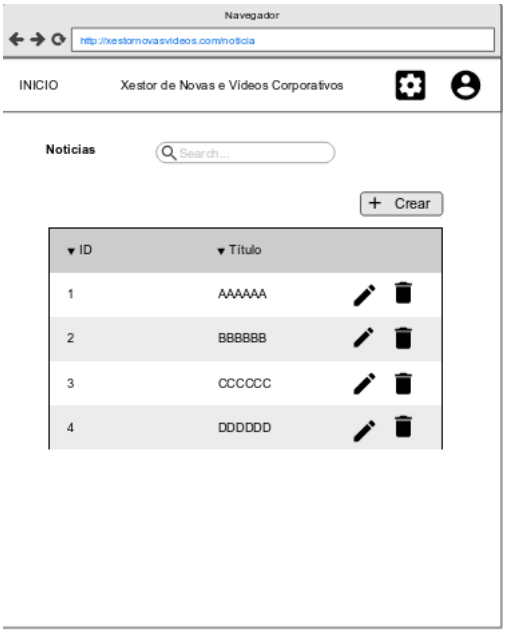

Figura 5.6: Mockup Administración de Noticias

Figura 5.7: Mockup Formulario Crear Vídeo

Figura 5.8: Mockup Formulario Editar Noticia

Capítulo 6
Implementación

En este apartado se va a tratar de explicar como se han desarrollado las funcionalidades de la plataforma, relacionados con la gestión y visualización de vídeos y noticias. Nos centraremos en los aspectos que han supuesto una mayor dificultad o son una parte muy importante de la implementación. Además de esto, se indicaran las medidas de seguridad implementadas, medidas de accesibilidad de la interfaz y cómo se ha llevado a cabo la internacionalización de la plataforma.

6.1 Vídeos

6.1.1 Gestión de vídeos

La función de la plataforma web que nos permite añadir vídeos al servicio fue una de las partes que más trabajo y problemas ha dado, principalmente por el uso de archivos de vídeo. El hecho de que se almacenen distintos tipos de vídeo también le dio mayor complejidad.

Para explicar la implementación que permite añadir vídeos, distinguiremos dos aproximaciones: por un lado el desarrollo que maneja archivos de vídeo almacenados localmente en el servidor, y por otro lado los vídeos obtenidos a partir de una URL de plataformas de vídeo.

Vídeos locales

Al añadir un vídeo es preciso obtener una miniatura en formato imagen que, o bien proporciona el usuario, o bien la crea automáticamente el servicio a partir del archivo. Para realizar esta acción se usa el siguiente código:

```
1  try(FFmpegFrameGrabber g = new FFmpegFrameGrabber(videoAux)) {
2      String nombreFichero = video.getTitulo().replace(" ", "_");
3      if (g != null) {
4          g.start();
5          OpenCVFrameConverter.ToIplImage converter = new
       OpenCVFrameConverter.ToIplImage();
6          g.setFrameNumber(10);
7          Frame frame = g.grabImage();
8          opencv_core.IplImage image = converter.convert(frame);
9          BufferedImage bi = iplImageToBufferedImage(image);
10
11         output = new File(rutaFicherosMiniaturas + nombreFichero +
       ".png");
12         int i = 1;
13         String nombreFicheroAux = nombreFichero;
14         while (output.exists()) {
15             nombreFichero = nombreFicheroAux + "_" + i;
16             output = new File(rutaFicherosMiniaturas +
       nombreFichero + ".png");
17             i++;
18         }
19         output.createNewFile();
20         ImageIO.write(bi, "png", output);
21         g.stop();
22     } else {
23         // Lanza error
24     }
25     miniaturaAux = new
       File(rutaFicherosMiniaturas+nombreFichero+".png");
26     video.setMiniatura(nombreFichero + ".png");
27 }
28
29 private BufferedImage iplImageToBufferedImage(opencv_core.IplImage
       src) {
30     OpenCVFrameConverter.ToIplImage grabberConverter = new
       OpenCVFrameConverter.ToIplImage();
31     Java2DFrameConverter paintCOnverter = new
       Java2DFrameConverter();
32     Frame frame = grabberConverter.convert(src);
33     return paintCOnverter.getBufferedImage(frame, 1);
34 }
```

Lo que se hace es lo siguiente. Se crea un objeto FFmpegFrameGrabber, que guarda el conjunto de frames del vídeo que se le pasa como parámetro. Se selecciona un frame, en nuestro caso el número 10 (línea 6 del código), a partir del cuál se genera una imagen en formato

png, a la cual se le asigna el nombre del vídeo. Antes de crear la imagen, se comprueba que no exista en la carpeta de las miniaturas un fichero con el mismo nombre, para evitar que se sobrescriban las existentes.

Otro aspecto a tener en cuenta a la hora de añadir un vídeo local, es su formato. Existe un conjunto de formatos que la aplicación es capaz de procesar. En caso de que el formato sea MP4, OGG o WEBM no es necesario realizar nada. Si es de un tipo diferente, el programa se encarga de transformarlo a un archivo MP4. Esto se debe a que el reproductor de HTML no es capaz de reproducir vídeos que no sean de los formatos anteriormente mencionados.

Para realizar esta acción, se usan dos implementaciones [3] diferentes:

- Librería JAVE

```
private void convertirVideo(File source, File target) throws
    ModelException {
    AudioAttributes audio = new AudioAttributes();
    audio.setCodec("aac");
    audio.setBitRate(64000);
    audio.setChannels(2);
    audio.setSamplingRate(44100);

    VideoAttributes video = new VideoAttributes();
    video.setCodec("h264");
    video.setX264Profile(VideoAttributes.X264_PROFILE.BASELINE);
    video.setBitRate(160000);
    video.setFrameRate(15);
    video.setSize(new VideoSize(400, 300));

    EncodingAttributes attrs = new EncodingAttributes();
    attrs.setFormat("mp4");
    attrs.setAudioAttributes(audio);
    attrs.setVideoAttributes(video);

    try {
        Encoder encoder = new Encoder();
        encoder.encode(new MultimediaObject(source), target, attrs);
    } catch (Exception e) {
        // Lanza error
    }
}
```

- Librería Xuggle

```
private void convertirVideoXuggle(File source, File target) {
    IMediaReader mediaReader =
    ToolFactory.makeReader(source.getAbsolutePath());
    IMediaWriter mediaWriter =
    ToolFactory.makeWriter(target.getAbsolutePath(), mediaReader);
    mediaReader.addListener(mediaWriter);
    while (mediaReader.readPacket() == null) ;
}
```

El hecho de usar dos implementaciones se debe a la realización de pruebas y de ver los resultados obtenidos con ambas librerías. Para algunos formatos de vídeo, la librería Xuggle devolvía vídeos con mejor calidad que la librería JAVE. Es por esto que a la hora de querer añadir un nuevo formato para que el servicio lo procese, se deben realizar pruebas con vídeos de ese formato para comprobar que implementación es más correcta para dicho formato.

Vídeos de plataformas

Para los vídeos de este tipo se deben extraer todos los datos posible del propio vídeo que esta subido en la plataforma. Las plataformas que han sido seleccionadas proporcionan una API a través de la cuál obtenemos estos datos. Los datos que se extraen en este caso son el título del vídeo y la URL de la miniatura.

La codificación de este proceso son las siguientes:

```
String url = "https://vimeo.com/api/v2/video/" + id + ".json";
ResponseEntity<String> res;
try{
    res = restTemplate.getForEntity(url, String.class);
}catch (HttpClientErrorException e){
    // Lanza error
}

if(res.getStatusCodeValue()!=200){
    // Lanza error
}

try{
    String jsonCuerpo = res.getBody().substring(1,
    res.getBody().length() - 1);
    JSONObject jsonVideo = new JSONObject(jsonCuerpo);
    video.setTitulo(jsonVideo.getString("title"));
    video.setMiniatura(jsonVideo.getString("thumbnail_large"));
    video.setVideo("https://vimeo.com/"+id);
} catch (Exception e){
```

```
    // Lanza error
}
```

```
String url = "https://www.googleapis.com/youtube/v3/videos?id=" +
    id + "&key=" + apiKey + "&part=snippet";
HttpEntity<String> res;
try{
    res = restTemplate.getForEntity(url, String.class);
}catch (Exception e){
    // Lanza error
}

JSONObject jsonVideo;
String titulo = "";
try {
    jsonVideo = new JSONObject(res.getBody());
    titulo = jsonVideo.getJSONArray(ITEMS).getJSONObject(0)
    .getJSONObject("snippet").getString("title");
} catch (Exception e){
    // Lanza error
}

return titulo;
```

En ambos casos vemos como se realiza una petición a la API correspondiente. Una vez obtenido la respuesta en formato JSON, se obtienen los valores deseados.

6.1.2 Visualización de vídeos

Este apartado es la parte más importante de la aplicación y también las más compleja, ya que se trata de algo para lo que no existe una implementación especifica ya creada.

Para la visualización de vídeos de múltiples fuentes, se han usado tres elementos clave:

- Elemento <video> de HTML
- YouTube Player API
- Vimeo Player SDK API

Ambas APIs estan desarrolladas en el lenguaje JavaScript. Lo que nos aportan estas APIs, a diferencia de los Iframes embebidos, es la posibilidad de interactuar, manejar y de escuchar eventos de los vídeos que se cargan en los reproductores creados por estas APIs, de un modo similar a lo que podemos realizar con el elemento <video> con los vídeos locales. Esto tiene

mucha importancia, ya que es lo que nos va a permitir implementar un carrusel de vídeos de todo tipo, de modo que cuando un vídeo termine, se pase al siguiente y comience a reproducirse.

El funcionamiento es el siguiente:

1. Se realiza la petición al servidor y se obtiene el listado de vídeos que se van a reproducir.

2. Se crean dos arrays vacíos, en los cuales se insertarán los objetos reproductores para vídeos de Youtube y Vimeo. Se crean también tres contadores con valor inicial 0, uno para cada tipo de vídeo, que servirán para referenciar posteriormente los reproductores dentro del array.

3. Se recorre la lista de vídeos obtenida. Por cada vídeo se crea un elemento <video> (LOCAL) o (YOUTUBE y VIMEO) en el elemento del carrusel, formando en el un listado. Los nombre de estos elementos insertados deben ser:

   TIPO_VIDEO-player-ContadorDeTipoVideo

   Por ejemplo si tenemos 2 vídeos locales y 2 de vimeo:

   LOCAL-player-0
   LOCAL-player-1
   VIMEO-player-0
   VIMEO-player-1

   Si el vídeo es de tipo YOUTUBE o VIMEO, se genera un reproductor con su API y se guarda guarda en el array en el orden que se ha recorrido la lista de vídeos. Es decir, el orden de los vídeos de Vimeo en la lista de vídeos debe ser el mismo que la de sus reproductores en el array.

4. Una vez creado el carrusel con todos sus vídeos, obtenemos el primer elemento y los reproducimos.

5. Cuando finaliza, se captura el evento de finalización. Una vez llega, se obtiene a través de JQuery el proximo elemento de vídeo. Una vez lo tenemos, obtenemos el identificador de dicho elemento, que va a ser el valor que le hemos dado anteriormente, el cual nos va a indicar que tipo de vídeo es:

   - Si tipo = ”LOCAL”, el elemento será un elemento <video>, y podremos ejecutar directamente con JQuery la función para reproducir el vídeo: &(id).play();

- Si tipo = ”VIMEO” o ”YOUTUBE”, necesitaremos obtener del propio id el valor del respectivo contador que se le había asignado. Este valor nos indica cual es el reproductor de este vídeo en el array, por lo que sólo se necesita obtener ese objeto y ejecutar la función de play propia de la API. (Este proceso se sigue también a la hora de ejecutar el primer vídeo en el paso 4)

6. Por último, se realiza el efecto de deslizado en el carrusel para que se muestre el vídeo que acaba de comenzar.

Otro aspecto relacionado con la implementación del carrusel es la vista de la imágenes que se ven debajo de los vídeos. En las implementaciones de carrusel típicas, los elementos situados debajo de lo que está mostrando el carrusel sirven para poder navegar por los distintos elementos del mismo.

En nuestro caso, estos elementos que se ven son las miniaturas, pero no queremos que tenga la función de navegación entre elementos. Como las imágenes deben tener un tamaño mínimo para que se puedan ver correctamente, no se pueden ver todas a la vez.

Lo que hemos implementado en nuestro carrusel es lo siguiente, en función del numero de miniaturas/vídeos que haya en cada momento:

- Si solamente se carga un vídeo, se va a mostrar su miniatura de manera centrada todo el rato.
- Si se cargan dos vídeos, se mostraran siempre ambas miniaturas. La miniatura del vídeo no se este mostrando se verá con un tono más apagado y la otra con un tono normal, dando ese efecto de que es el vídeo activo.
- Si se cargan 3 o 4 miniaturas, se visualizan 3 imágenes estando en el centro la miniatura del vídeo actual, a su derecha la miniatura del próximo vídeo y a su izquierda la miniatura del vídeo anterior. En caso de ser el primer vídeo, se ve la miniatura del último vídeo de la lista.
- Si se cargan 5 o más vídeos, se visualiza de un modo similar al anterior punto, pero ahora se ven las miniaturas de los dos siguientes vídeos y las de los dos anteriores.

Para realizar esto, lo que se hizo fue ocultar los indicadores que se generan inicialmente en el carrusel, y se crean las imágenes iniciales que se necesiten. Una vez que acaba un vídeo, se realiza una acción, de nuevo, en función del número de vídeos:

- Si solo hay una miniatura, no se realiza ninguna acción

- Si hay dos miniatura, se marca la imagen de otro vídeo como activa y la que estaba activa se elimina la clase.
- Si hay 3 miniaturas, se elimina de la lista la miniatura de la izquierda y se añade una nueva imagen por la derecha. De este modo, queda en el centro la imagen del vídeo que se va a comenzar a reproducirse, la cual se marca como activa.
- Si se cargan 5 o más miniaturas, se realiza lo mismo que en el caso anterior.

6.2 Noticias

La gestión de noticias es mucho más sencilla. Los cuerpos de las noticias pueden ser muy extensos, por lo que en vez de almacenarlos en la base de datos como un campo más, se almacenan en ficheros dentro del servidor.

El único problema que se puede considerar respecto a las noticias, fue como gestionar los saltos de línea que definen los párrafos del cuerpo, ya que en el contenido de un JSON no puede haber estos saltos de línea. Para solucionar esto, se sustituyen los saltos de línea por el texto "<br >", que es el elemento que se usa en HTML para realizar un salto de línea.

Cuando se accede a la pantalla donde se muestra el contenido de la noticia, se obtiene su cuerpo y se divide su texto por cada elemento "<br >" encontrado. Con cada uno de los textos obtenidos se genera un párrafo, y así conseguimos que los párrafos de las noticias se visualicen correctamente.

Es posible que surja la duda de por qué no se añade el contenido del cuerpo de la noticia al documento HTML, ya que si el texto ya cuenta con los elementos <br>, los saltos de línea ya se visualizarían correctamente. Esto no se realiza así por una medida de seguridad que se comentará más adelante.

6.3 Internacionalización

Todos los elementos de la interfaz han sido internacionalizados. Para lograrlo, se ha creado un fichero por cada uno de los lenguajes aceptados, los cuales contienen un listado de {identificador,valor}, donde los identificadores se repiten en todos los ficheros, y los valores varían en función del idioma. En estos ficheros se guardan los textos que veremos después en la pantalla, como las cabeceras de las tablas, etiquetas de los botones, campos de formularios,etc. Los nombres de los ficheros son es.js, en.js y gl.js.

Cada vez que se carga una página de la aplicación, se obtiene del servidor el valor del idioma actual (es, en o gl). Con este valor, se carga el fichero correspondiente. Para obtener los valores, se implementa una función a la cual se le pasa un identificador, y esta accede a la lista para devolver el texto correspondiente. Para que esto funcione correctamente, todos los ficheros deben generar el mismo objeto, para poder hacer referencia a él, independientemente del fichero cargado.

6.4 Seguridad

Autenticación

Como ya sabemos, una parte de la plataforma web debe ser accesible únicamente para los administradores. Estas páginas deben estar protegidas con autenticación.

Spring Security es una herramienta de Spring muy útil, que nos permite indicar que páginas deben estar autenticadas y cuales no, además de implementar una propia gestión de usuarios, la cual hemos modificado para poder almacenar los usuarios en nuestra BD. [4]

```
http.
requiresChannel()
.anyRequest()
.requiresSecure()
.and()
.authorizeRequests()
.antMatchers("/inicio","/").permitAll()
.antMatchers(resources).permitAll()
.antMatchers(HttpMethod.GET,"/noticias","/videos").permitAll()
.antMatchers("/login").permitAll()
.anyRequest().authenticated()
.and()
.formLogin()
.loginPage("/login").failureUrl("/login?error=true")
.defaultSuccessUrl("/admin")
.usernameParameter("user_name")
.passwordParameter("password")
.and()
.logout()
.logoutUrl("/logout")
.deleteCookies("JSESSIONID", "SESSION")
.invalidateHttpSession(true)
.logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
.logoutSuccessUrl("/inicio")
.and()
.csrf();
```

En el código podemos ver la configuración de Spring Security en nuestro proyecto. La norma, por defecto, es proteger todas las páginas de nuestra plataforma. Las que queremos que sean abiertas se deben indicar específicamente. Se puede observar como se indica la página de inicio, junto con las llamadas al servicio que devuelven los vídeos y noticias que se van a mostrar en el portal de cara al usuario.

El método de autenticación escogido es ”Login Form”, de manera que si queremos acceder a una página protegida y no hemos iniciado sesión, la aplicación nos redirige a una página de login.

OWASP-10

Existen muchos ataques conocidos realizados contra aplicaciones web. Para nuestra plataforma, se ha buscado información sobre algunos de ellos.

En OWASP Top 10 [5, 6] se enumeran todos los años los riesgos de seguridad más importantes. Para securizar nuestra aplicación, se han seguido las medidas y recomendaciones para cada uno de estos riesgos:

- Habilitar HTTPS
- SQL injection
  - Uso de ORM, evitando el uso directo del interprete (uso de JPA/Hibernate)
  - Parametrizar consultas en vez de concatenar cadenas de texto y convertir los parámetros a su tipo de valor esperado (Integer, Booleano,...)

```
1  public void remove(Video video) throws ModelException {
2      if(video.getId()==null){
3          throw new ModelException(ExceptionType.INTERNO,"El video no
       existe en la BD");
4      }
5      sessionFactory.getCurrentSession().delete(video);
6
7  }
8  public Video findById(Long id) {
9      Query query =
       sessionFactory.getCurrentSession().createQuery("from Video where
       ID_VIDEO = :idVideo");
10     query.setParameter("idVideo",id);
11     return  (Video) query.uniqueResult();
12 }
```

En el código anterior podemos ver un ejemplo de los puntos anteriores. En el método remove se usa la clase SessionFactory de Hibernate, la cual proporciona un

método específico para eliminar un elemento de la BD, sin necesidad de escribir a mano la consulta. En el método findById, no se concatenan caracteres, sino que se parametriza la consulta y se le pasa un valor del tipo deseado, en este caso tipo Long.

  - En caso de error en la BD, no se muestra al usuario el error especifico, se muestra simplemente error interno.
  - Rechazar caracteres extraños
  - Filtrar y validar los valores en los inputs de los formularios

- Pérdida de autenticación
  - Se cifran datos de inicio de sesión através de HTTPS
  - No se muestran datos de inicio de sesión en la URL. Al realizar el login, los datos van en el cuerpo de la petición, los cuales se cifran con el uso de HTTPS.
  - La sesión caduca tras X tiempo de inactividad
  - Desactivar usuarios tras 3 intentos fallidos [7]. Cada usuario cuenta con un valor en la base de datos llamado Intentos. Este indica el número de veces que se ha tratado de iniciar sesión con ese usuario, y la contraseña no ha sido la correcta. Si llega a 3, el usuario es bloqueado.

    En el método de autenticación de spring security, se ejecuta el siguiente código en caso de que las credenciales sean incorrectas:

```
int cuenta = usuario.getIntentos()+1;
usuario.setIntentos(cuenta);
if(cuenta >= MAX_INTENTOS) {
    usuario.setActivo(false);
    isLocked = true;
}
try{
    userDao.update(usuario);
}
```

    Podemos ver que se comprueba si el número de intentos fallidos es igual al máximo permitido. Si se cumple, se desactiva el usuario y se actualiza en la base de datos.

- XSS
  - Validar y limitar tamaño inputs en formularios

  - Eliminar caracteres especiales
  - Spring security por defecto añade cabecera de protección XSS
  - A la hora de escribir textos en los componentes de la interfaz, JQuery cuenta con un método en el cual añade el texto como tal, en vez de añadirlo como contenido HTML, lo cual evita problemas a la hora de insertar datos. Esta medida es la que se hacía referencia en el apartado de gestión de noticias.

    Para añadir el texto de las noticias a su contenedor se usa $("#idContenedor").text(cuerpoNoticia), en lugar de $("#idContenedor").html(cuerpoNoticia)

- Revisión de dependencias del proyecto, uso de versiones recientes y compatibles entre ellas.

- CSRF

  Todas las llamadas que necesiten autenticación realizadas desde la interfaz se añade la cabecera "X-header-csrf", cuyo valor es el token de autenticación. Con esto se consigue evitar que se realicen peticiones no deseadas desde páginas externas.

  Spring security permite activar el control de CSRF, encargándose de devolver un error 403 en caso de detectarse un ataque de este tipo.

Sonar

Para mejorar la calidad del código de la aplicación se ha usado la versión gratuita de Sonar [8]. Además de esta funcionalidad, también es capaz de detectar fallas en el código e implementaciones que pueden dar problemas en el futuro. Algunas de las medidas que se han implantado tras los análisis de sonar son:

- Cerrar y liberar siempre los ficheros: sonar detecta en el código cuando se hace uso incorrecto de ficheros, y podían quedar bloqueados

- Protección ante ataque ReDos: a la hora de añadir un vídeo de Vimeo o Youtube se valida la url con un patrón REGEX. Este ataque de denegación de servicio se da cuando en un patrón existe una parte en la cual se comprueba el caso "1 o más" o "0 o mas", ya que REGEX comprueba todos los casos posibles. El ejemplo típico para explicar es este:

  (a+)+ ⟶ si se usa "aa" valida 4 casos, si se pasa "aaaa" valida 16 casos y asi sucesivamente

  Para evitar esto, lo que hice fue eliminar este tipo de comprobaciones en las expresiones regulares utilizadas, haciendo uso de expresiones que sean de una simple comprobación, o en caso de no poder hacerlo, ponerle un limite al numero de repeticiones que se puedan dar como máximo.

[http|https]+://(?:www.|)vimeo.com/([a-zA-Z0-9_-]+)(&.+)?
https?://(?:www.|)vimeo.com/([a-zA-Z0-9_-]{1,10})

En este caso por ejemplo, se elimina ([a-zA-Z0-9_-]+), y se cambia por [a-zA-Z0-9_-]{1,10}.

6.5 Accesibilidad

Los aspectos de accesibilidad más destacados son los siguientes:

- Todos los elementos con los que puede interactuar el usuario son accesibles a través del teclado, y se le ha asignado la funcionalidad cuando se pulsa el botón de ENTER sobre ellos. Algunos elementos de HTML ya permiten navegar por ellos mediante el uso del teclado, como son los botones o los enlaces. Para los casos que no admiten esto por defecto, se ha añadido el atributo ”tabindex=0”. De este modo podemos acceder a él pulsando el tabulador.
- Uso de subtítulos en el reproductor. Se habilita un botón que activa o desactiva los subtítulos en todos los objetos de reproductores que se han creado. De nuevo, debemos distinguir esta funcionalidad para cada tipo de vídeo:
  - Vimeo

```
this.playersVimeo.forEach(function(player) {
 player.enableTextTrack(Application.LANGUAGE).then(function(track) {
     }).catch(function(error) {
      switch (error.name) {
          case 'InvalidTrackLanguageError':
              player.enableTextTrack("en");
              break;
          case 'InvalidTrackError':
              // There is no such text track
              break;

          default:
              break;
      }
    });
}
```

Se recorre la lista de reproductores de vídeo de Vimeo y se activan los subtítulos con el método enableTextTrack. Como parámetro se le pasa el código del lenguage de la aplicación. Si tiene subtítulos para ese idioma se muestran. En caso de que no tenga para ese idioma se prueba con los subtítulos en inglés.

- Youtube

```
this.playersYoutube.forEach(function(player) {
    player.setOption('cc_load_policy', 1);
    player.loadModule('captions');
    player.setOption('captions', 'track', {languageCode:
    Application.LANGUAGE});
});
```

Se recorre la lista de reproductores de vídeo de youtube y se aplica añaden las siguientes opciones, pasando como parámetro el lenguaje de la aplicación.

En el caso de YouTube, algunos vídeos cuentan con sus propios subtítulos. Si existe para el idioma de la aplicación éstos se muestran. Si el vídeo no tiene subtítulos propios, se utilizan los subtítulos autogenerados que ofrece Youtube, aunque en este caso solo se ha conseguido implementar que se muestren los subtítulos si el idioma en el que se habla en el vídeo es el mismo que el idioma de la aplicación.

- Local

  No se ha implementado. (Ver punto 1 de la sección 9.2)

- Uso de roles de WAI-ARIA [9, 10, 11, 12] en todos los elementos de la interfaz

Capítulo 7

Pruebas

Para validar el buen funcionamiento de la aplicación se han realizado pruebas de integración, unitarias, de seguridad y de accesibilidad.

Unitarias

Las pruebas unitarias son aquellas son aquellas que nos indican que un fragmento de código funciona correctamente. Estas pruebas se han realizado para las clases DAO, Servicios y la clase Controlador.

Para realizar estas pruebas se ha usado la librería Mockito, que nos permite sustituir elementos del código que hacen llamadas a clases externas que para estas pruebas no son necesarias usar, haciendo que nos centremos exclusivamente en la clase a testear. Un ejemplo de esto se da en la clase Service, en el cual se realizan llamadas a funciones del objeto DAO. Lo que se hace es controlar el comportamiento de este objeto, de manera que simule que se esta usando esa función y devuelva el resultado deseado para cada caso, pero en realidad no se está ejecutando la función. De este modo se consiguen hacer tests independientes y unitarios.

Integración

Una vez probados todas las capas del código por separado, se realizan tests para comprobar que funcionan juntas correctamente.

Para estas pruebas se usa la librería JUnit. Requiere también una serie de ficheros de prueba, como ficheros de vídeo y miniaturas, para los casos de creación de vídeos. Para que no afecten a los datos de la aplicación, se ha creado un nuevo esquema en la base de datos que contiene las mismas tablas que el original (Vídeo, Noticia y Usuario), sobre las que se van a insertar, editar o eliminar datos en las pruebas.

Se han realizado pruebas del servicio a través de Postman, donde podemos observar de un modo más claro las respuestas que devuelve el servidor.

Seguridad

- Para comprobar el correcto funcionamiento de detección de ataques csrf, se ha deshabilitado el control del token de autenticación en spring security. Una vez hecho esto, se genera una página falsa, desde la cual se va a realizar una petición a nuestra aplicación (Aclaración: previo a esto se debe haber iniciado sesión con un usuario en la aplicación).

  En un fichero se crea una página ”maliciosa”, cuyo contenido es el siguiente:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org" lang="en">
<head>
    <meta charset="UTF-8">
    <title>Actual Application</title>
</head>
<body>
    <form method="post" action="https://localhost:8443/logout">
    <input type="submit" value="Logout" />
    </form>
</body>
</html>
```

  Si pulsamos el botón se realiza la acción del formulario. Como está desactivada la protección CSRF, se va a ejecutar con normalidad. Si volvemos a la página normal y tratamos de hacer cualquier acción con nuestro usuario, nos volverá a la página de login, ya que se ha cerrado la sesión desde la pagina maliciosa.

- Para probar la protección frente ataques XSS, se ha probado a insertar código Javascript en el título de una noticia. Cuando se carga la página principal vemos como se ejecuta el código (Figura 7.1).

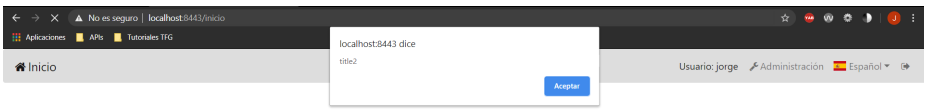

Figura 7.1: Protección XSS desactivada

  Sin embargo, cuando se activa vemos como se muestra el texto correctamente, sin que sea código ejecutable, a pesar de tener la sintaxis de un script (Figura 7.2).

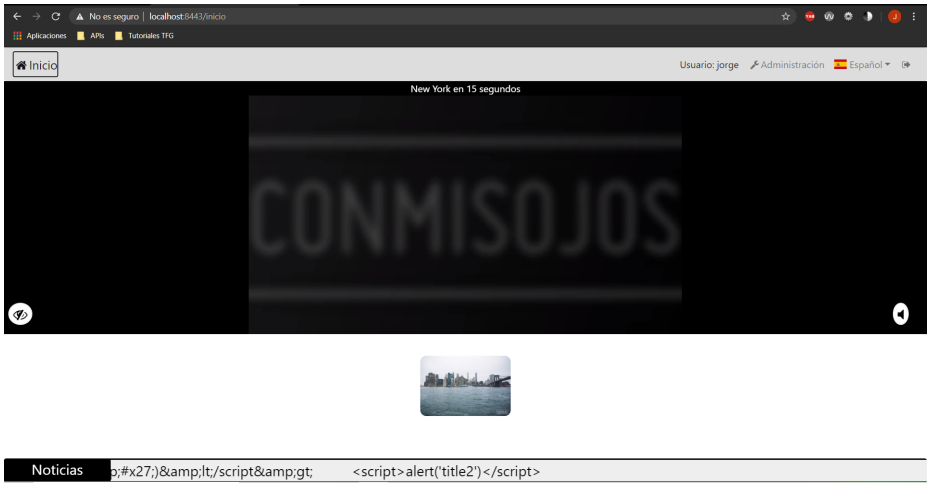

Figura 7.2: Protección XSS activada

Accesibilidad

Se ha probado a realizar todas las funcionalidades que ofrece la plataforma usando únicamente el teclado.

Capítulo 8

Manual de Usuario

En este apartado se describen las pantallas, menús y formularios de la aplicación, indicando el funcionamiento de cada uno de los elementos con los que puede interactuar el usuario.

Cabecera

La cabecera se muestra en la parte superior de todas las pantallas de la aplicación (Figuras 8.1 y 8.2.

Figura 8.1: Cabecera sin iniciar sesión

Figura 8.2: Cabecera con inicio de sesión

Los elementos de la cabecera son los siguientes:

- Inicio: enlace al portal web
- Usuario: muestra el nombre del usuario con el que se ha iniciado sesión
- Administración: enlace a la pantalla de administración. Si previamente no se ha iniciado sesión, redirige a la pantalla de login.
- Lenguaje: enlaces que permiten cambiar el lenguaje de la aplicación [13]
- Logout: cierra la sesión actual

Pantalla de Inicio

La pantalla de inicio es la pantalla del portal de la plataforma, donde se muestran los vídeos y las noticias a los usuarios (Figura 8.3).

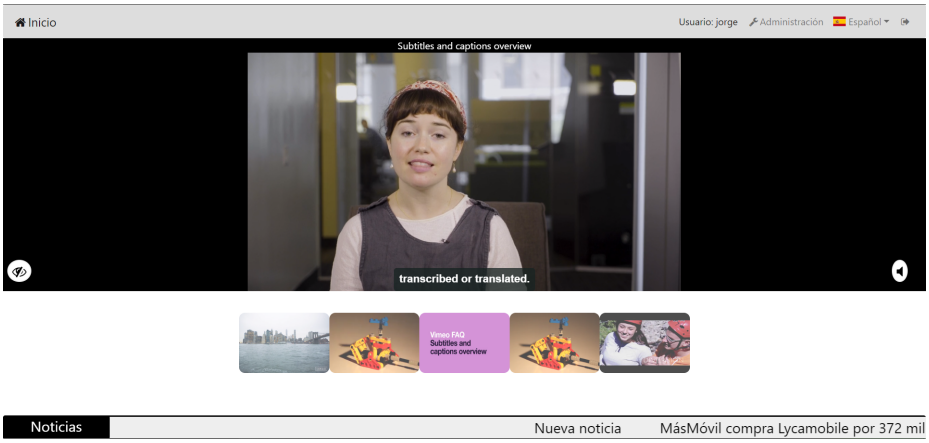

Figura 8.3: Pantalla de Inicio

Los elementos con los que puede interactuar el usuario son los siguientes:

- Subtítulos: activa o desactiva los subtítulo de los vídeos
- Volumen: activa o desactiva el volumen del reproductor
- Títulos de noticias: enlaces a la noticia correspondiente al título

Pantalla de Noticia

Pantalla en la que se muestra el contenido de una noticia (Figura 8.4).

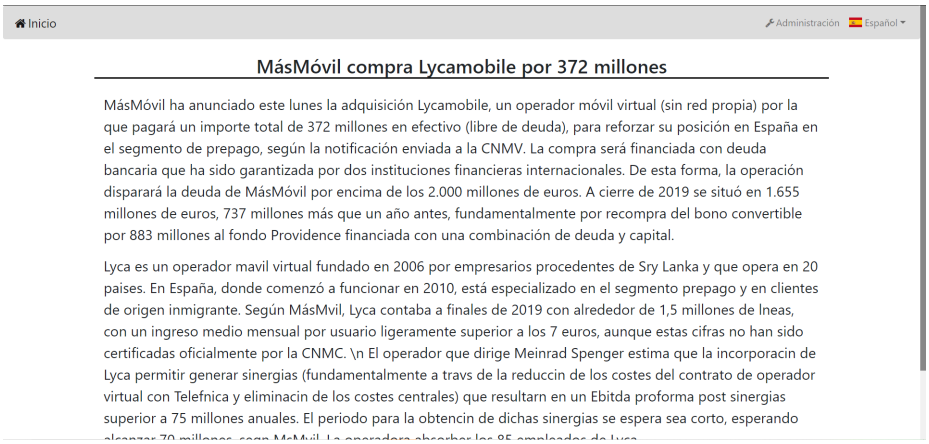

Figura 8.4: Pantalla de noticia

Menú de Administración

Menú de acceso a las pantallas de administración (Figura 8.5).

Figura 8.5: Menú de administración

Pantalla de Administración de Vídeos

En esta pantalla se muestra una tabla con el listado de todos los vídeos que han sido guardados en el servidor (Figura 8.6). Los elementos con los que puede interactuar el usuario son los siguientes:

Figura 8.6: Pantalla de administración de vídeos

- Filtro título: Campo de texto que permite filtrar los resultados de la tabla por el campo de título. Cada vez que se modifique el contenido de este campo, se actualiza la tabla con los vídeos cuyo título contiene la cadena escrita.

- Filtro fecha: Campo que permite filtrar los resultados de la tabla por los campos fecha de inicio y fecha de fin. Cuando se selecciona una fecha en este campo, se actualiza la tabla con los vídeos cuya fecha de inicio es anterior a la fecha indicada y cuya fecha de fin es nula o posterior a la fecha indicada.

- Botón ”Crear”: Abre un diálogo que contiene un formulario para añadir un nuevo vídeo al servidor (Figura 8.7).

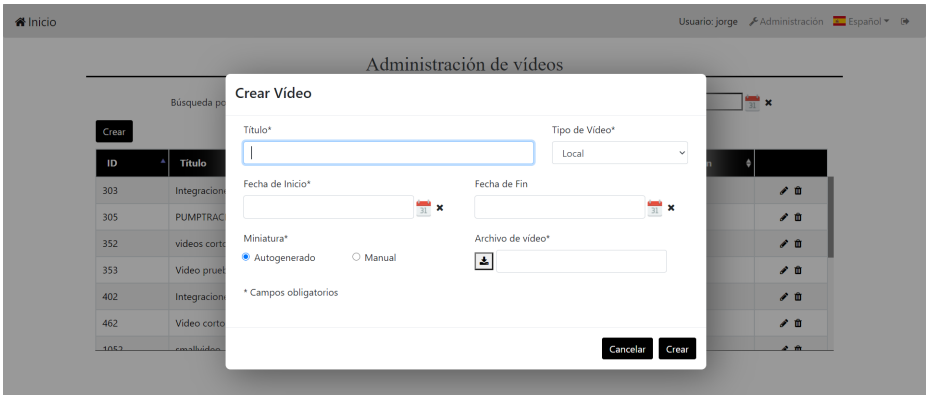

Figura 8.7: Formulario de vídeos

- Icono ”Lápiz”: Abre un diálogo que contiene el mismo formulario mencionado en el punto anterior, con la diferencia que los campos van a estar completados con los valores actuales del vídeo seleccionado. Permite editar la información de un vídeo.

- Icono ”Papelera”: Permite borrar del servidor el vídeo seleccionado. Muestra previamente un diálogo de confirmación.

Pantalla de Administración de Noticias

En esta pantalla se muestra una tabla con el listado de todas las noticias que han sido guardadas en el servidor (Figura 8.8).
Los elementos con los que puede interactuar el usuario son los siguientes:

Figura 8.8: Pantalla de administración de noticias

- Filtro título: Campo de texto que permite filtrar los resultados de la tabla por el campo de título. Cada vez que se modifique el contenido de este campo, se actualiza la tabla con las noticias cuyo título contiene la cadena escrita.
- Filtro fecha: Campo que permite filtrar los resultados de la tabla por los campos fecha de inicio y fecha de fin. Cuando se selecciona una fecha en este campo, se actualiza la tabla con las noticias cuya fecha de inicio es anterior a la fecha indicada y cuya fecha de fin es nula o posterior a la fecha indicada.
- Botón "Crear": Abre un diálogo que contiene un formulario para añadir una nueva noticia al servidor (Figura 8.9).

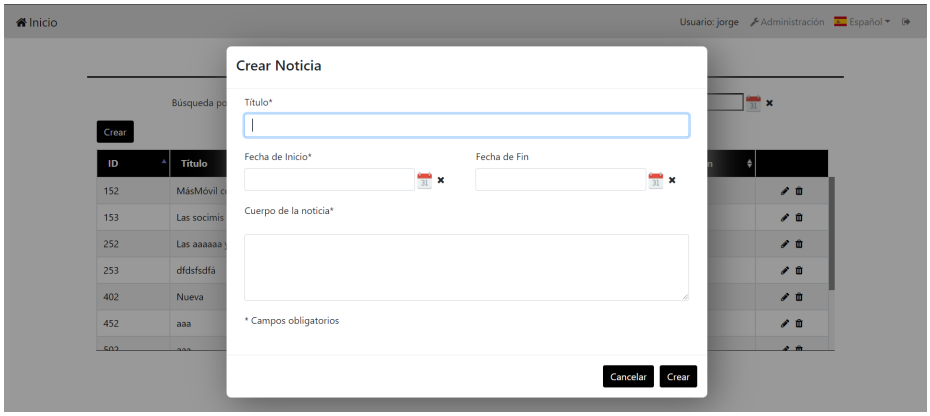

Figura 8.9: Formulario de noticias

- Icono "Lápiz": Abre un diálogo que contiene el mismo formulario mencionado en el punto anterior, con la diferencia que los campos van a estar completados con los valores actuales de la noticia seleccionada. Permite editar la información de una noticia.
- Icono "Papelera": Permite borrar del servidor la noticia seleccionado. Muestra previamente un diálogo de confirmación.

Pantalla de Administración de Usuarios

En esta pantalla se muestra una tabla con el listado de todos los usuarios que han sido guardados en el servidor (Figura 8.10).
Los elementos con los que puede interactuar el usuario son los siguientes:

Figura 8.10: Pantalla de administración de usuarios

- Filtro nombre: Campo de texto que permite filtrar los resultados de la tabla por el campo de nombre. Cada vez que se modifique el contenido de este campo, se actualiza la tabla con los usuarios cuyo nombre contiene la cadena escrita.
- Botón ”Crear”: Abre un diálogo que contiene un formulario para añadir un nuevo usuario al servidor (Figura 8.11).

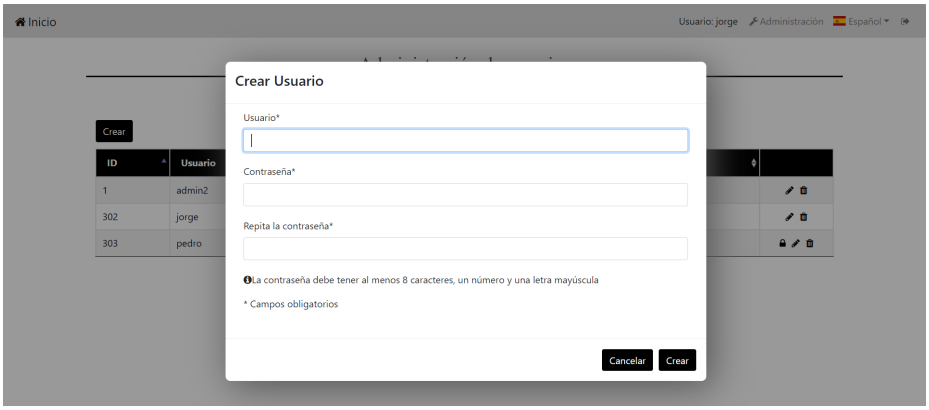

Figura 8.11: Formulario de usuarios

- Icono ”Lápiz”: Abre un diálogo que contiene un formulario que permite modificar la contraseña de un usuario.
- Icono ”Papelera”: Permite borrar del servidor el usuario seleccionado. Muestra previamente un diálogo de confirmación.

Capítulo 9

Conclusiones

Una vez finalizado el proyecto, se analizan que objetivos se han cumplido y se da una opinión personal acerca del trabajo realizado.

9.1 Conclusiones

Los objetivos marcados antes de comenzar el proyecto se han cumplido. Se ha implementado un gestor que nos proporciona las funcionalidades necesarias para administrar las noticias y los vídeos, y que se muestren en el portal tanto unos como otros, sin depender los vídeos de su origen o formato.

Gran parte del tiempo invertido en la aplicación ha sido para el desarrollo del reproductor y en documentarse sobre las APIs de reproductores de Youtube y Vimeo. Ambas cuentan con una documentación amplia y detallada, pero aun así hay muchas funcionalidades sobre las que fue difícil buscar información. Además algunos aspectos no funcionan siempre como está documentado, por lo que se tuvieron que realizar muchas pruebas para llegar a obtener un funcionamiento correcto.

Como aspecto negativo de estas APIs, es que las sensaciones tras realizar múltiples pruebas con ellas, es de no tener el control total sobre los vídeos. Ocurren problemas como que una vez se está repitiendo el vídeo en un reproductor, tras haber terminado antes, este vuelve a reproducirse pero solamente se escucha el sonido, y el reproductor se queda mostrando enlaces a vídeos recomendados, y a la próxima vez que vuelve a reproducirse si lo hace correctamente. Otro problema similar ocurre cuando sin razón alguna no se carga todo el vídeo, de modo que se queda pausado y no vuelve a reproducirse, deteniendo el funcionamiento del carrusel.

Otro motivo por el cual no usaría estas herramientas en una página web es por la estética. En páginas web de esta categoría es un aspecto fundamental. Cuando vemos un vídeo en la página de una empresa queremos que sea de calidad y claro, sin que aparezcan otros elementos en la vista del reproductor. Ninguna de las APIs permite eliminar los elementos característicos de sus reproductores, como el usuario del vídeo, la barra de reproducción... En nuestra implementación, podemos ver que cada vez que se inicia un vídeo de Vimeo o de Youtube se ven estos elementos, lo cuál rompe bastante la estética del portal web.

9.2 Líneas futuras

- El reproductor de vídeo permite mostrar subtítulo para vídeos de Vimeo y de Youtube, pero no para los vídeos almacenados en el servidor. Esto se debe a que los subtítulo no fue un tema que se consideró desde el principio y no se comenzó a trabajar con ellos casi hasta el final del desarrollo de la aplicación.

  En el caso de las APIs, era relativamente sencillo su uso, y únicamente se tuvo que añadir una nueva función en el comportamiento de la vista. Sin embargo, para el caso de los vídeos locales es más complejo. En el elemento <video> de HTML se permite añadir el componente <track>. En este componente se cargan archivos de formato VTT. Para realizar esto, se tendrían que añadir campos nuevos para las clases de vídeo, añadir nuevos campos en los formularios de creación de vídeo, modificar la entrada de las funciones del controlador y realizar pruebas. Además, si se quiere tener unos subtítulos correctos de cada vídeo local, habría que crearlos a mano, lo cual llevaría tiempo en documentarse y obtener herramientas para hacerlo, y sería mucho tiempo perdido.

  Por todo esto, se ha dejado está funcionalidad para posibles futuras versiones.

- Mejorar la estética del portal web. Con tiempo para conocer más sobre lenguaje CSS y pensar posibles diseños, podría darse un mejor aspecto a la pantalla inical, principalmente al listado de imágenes y al ticker de noticias.

Apéndices

Apéndice A

Material adicional

A.1 Scrum

Scrum [14] es un marco de trabajo para desarrollo ágil de software. Su objetivo es reducir la complejidad del desarrollo de productos, llevado a cabo en un entorno de trabajo colectivo, donde los requisitos cambian constantemente y los resultados se tienen que obtener en un plazo corto de tiempo.

Scrum promueve la colaboración y transparencia en los equipos. Esto se debe a que todos y cada uno de los componentes se comunican entre ellos, comparten la misma información y todos deben tener conocimiento en todos los aspectos del desarrollo.

Esta metodología se basa en una serie de roles, eventos y artefactos, se indican y se definen brevemente a continuación:

Roles

Dentro del equipo encargado de llevar a cabo el proyecto existen diferentes roles:

- Product Owner
  Se encarga de comunicarse y estar en contacto con el cliente. Gestiona el product backlog.
- Scrum Master
  Es el líder del equipo Scrum y forma parte del mismo. Se encarga de facilitar y ayudar en todos los problemas que puedan surgirle a los miembros del equipo Scrum, así como comunicarse con el Product Owner para trasladar dudas o problemas de algun miembro del equipo.

- Equipo Scrum
  Equipo de desarrolladores encargado de abordar las tareas definidas en los sprints.

Eventos

- Sprint
  Período de tiempo limitado con un máximo de dos semanas en el cual se deben abordar las tareas planificadas

- Scrum daily
  Reunión diaria con una duración aproximada de 15 minutos dirigida por el Scrum Master, en la cual participan todos los miembros del equipo Scrum. El objetivo de estas reuniones es indicar el estado actual de las tareas, informando cada uno de los miembros las tareas que ha realizado el día anterior, los problemas con los que se ha encontrado y las tareas planificadas para hoy.

- Revisión de sprint
  Una vez finaliza el sprint, el Equipo Scrum presenta al Product Owner el trabajo realizado durante el mismo. El Product Owner decide si lo entregado cumple o no con los requisitos acordados con el cliente.

- Retrospectiva
  Reunión similar a la revisión de sprint, pero esta vez solo participan los miembros del Equipo Scrum. También se realiza una vez finalizado el sprint. Se determina que fue lo que se ha llevado a cabo correctamente y lo que no, además de aclarar en que aspectos se puede mejorar al equipo en vistas a próximos sprints.

Artefactos

- Product Backlog
  Lista de tareas que describen los requisitos del proyecto.

- Sprint Backlog
  Lista de tareas extraídas del Product Backlog que se deben abordad en el Sprint.

- Incremento Conjunto de todas las tareas finalizadas desde al inicio hasta la última versión que se ha entregado al cliente. El Equipo Scrum es el responsable de que todo funciona correctamente antes de ser entregado, aunque es el Product Owner el que debe dar el visto bueno.

A.2 Tablero Kanban

El tablero Kanban es la herramienta para asociar y visualizar su flujo de trabajo y uno de los componentes claves del método Kanban. Originalmente, se utilizaba una pizarra blanca (o un tablero de corcho) que se dividía en columnas y filas.

Cada columna visualiza una fase de su proceso y las filas representan diferentes tipos de actividades específicas (diseño, errores, pruebas, etc.).

Al mismo tiempo, cada tarea que entra en su flujo de trabajo aparece en el tablero como una tarjeta Kanban. El primer estado de cada tarjeta es la columna "Por hacer". Hoy en día, existen soluciones de tarjetas Kanban digitales más prácticas y accesibles a nivel global que son perfectas tanto para para los equipos remotos como para los equipos que desarrollan su actividad en el lugar donde se realiza el proyecto.

Lista de acrónimos

API *Application Programming Interface*

URL *Uniform Resource Locator*

BD *Base de datos*

HTTPS *HyperText Transfer Protocol Secure*

JPA *Java Persistance API*

XML *Extensible Markup Language*

SQL *Structured Query Language*

AJAX *Asynchronous JavaScript And XML*

DAO *Data Access Object*

JSON *JavaScript Object Notation*

JAVE *Java Audio Video Encoder*

VTT *Video Text Tracks*

Glosario

Ticker Zona de la pantalla reservada para la visualización de titulares de noticias en formato de texto desplazándose horizontalmente de derecha a izquierda.

Frame Cada una de las imágenes instantáneas en las que se divide un vídeo.

Endpoint Punto de entrada de un servicio a través del cuál se realiza una petición.

Mockup Modelo o prototipo que se utiliza para exhibir o probar un diseño.

Bibliografía

[1] "Postgresql vs mysql ¿cuál usar para mi proyecto?" [En línea]. Disponible en: https://blog.mdcloud.es/postgresql-vs-mysql-cual-usar-para-mi-proyecto/

[2] "Las metodologías ágiles más utilizadas y sus ventajas dentro de la empresa." [En línea]. Disponible en: https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/

[3] "Convert video to another format in spring boot." [En línea]. Disponible en: https://medium.com/tekraze/convert-video-to-another-format-in-spring-boot-java-based-apps-7763fbc4d7ce

[4] "Srping boot security login + mysql." [En línea]. Disponible en: http://cristianruizblog.com/spring-boot-security-login/

[5] "Owaspguide." [En línea]. Disponible en: https://wiki.owasp.org/images/5/5e/OWASP-Top-10-2017-es.pdf

[6] "Owaspguide." [En línea]. Disponible en: https://owasp.org/www-project-top-ten/

[7] "Spring security : limit login attempts example." [En línea]. Disponible en: https://mkyong.com/spring-security/spring-security-limit-login-attempts-example/

[8] "Sonarscanner." [En línea]. Disponible en: https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/

[9] "Aria in html." [En línea]. Disponible en: https://www.w3.org/TR/html-aria/

[10] "Web content accessibility guidelines 2.1." [En línea]. Disponible en: http://romeo.elsevier.com/accessibility_checklist/

[11] "Wai-aria, una aproximación." [En línea]. Disponible en: http://www.nosolousabilidad.com/articulos/wai_aria.htm

[12] "Accessible rich internet applications (wai-aria) 1.1." [En línea]. Disponible en: https://www.w3.org/TR/wai-aria-1.1/#attrs_liveregions

[13] "Bootstrap 4 multilingual site menu." [En línea]. Disponible en: https://forum.pinegrow.com/t/bootstrap-4-multilingual-site-menu/1720

[14] "Scrum." [En línea]. Disponible en: https://beagilemyfriend.com/scrum/